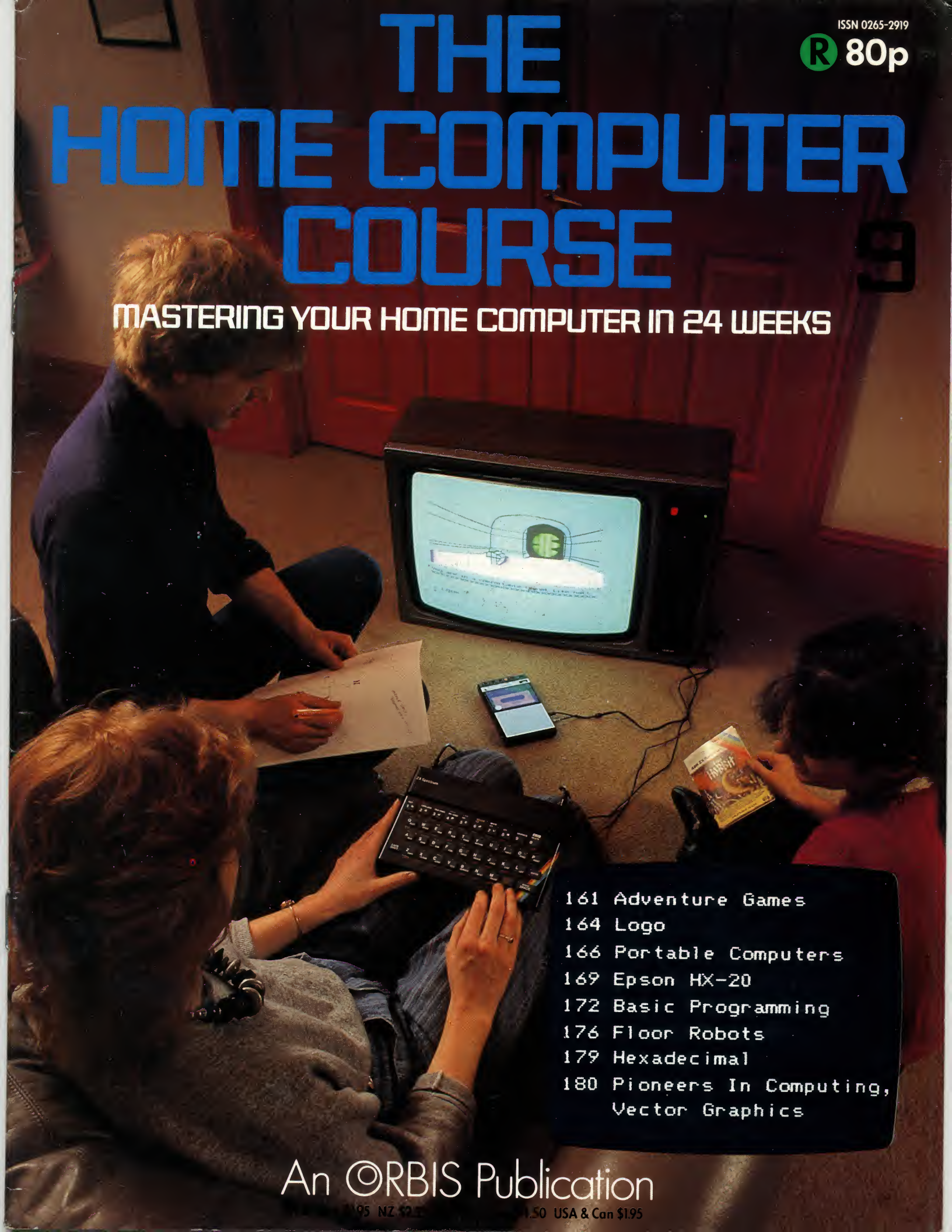
ISSN 0265-2919

80p

THE HOME COMPUTER COURSE

9

MASTERING YOUR HOME COMPUTER IN 24 WEEKS



161 Adventure Games
164 Logo
166 Portable Computers
169 Epson HX-20
172 Basic Programming
176 Floor Robots
179 Hexadecimal
180 Pioneers In Computing, Vector Graphics

An ORBIS Publication

95 NZ $2.2 1.50 USA & Can $1.95

CONTENTS

Hardware

The Age Of Portables Computers are now becoming so small they can be carried in a briefcase. But what would you use such a device for? 166

Epson HX-20 The first genuinely portable computer to feature a built-in display, printer and microcassette 169

Software

Adventure Playground Adventure games are now rivalling arcade games in popularity. What makes them so addictive? 161

Basic Programming

Leaving It To Chance Producing random numbers can be quite a difficult task for a computer, but they're needed for games 172

Insights

Microworlds Limited environments are set up on a computer using the Logo language, in which children can explore and learn 164

On Two Wheels A floor robot can make both an entertaining and educational add-on to your home computer 176

Passwords To Computing

Sweet Sixteen Counting in base 16 can be more productive than you might think 179

Pioneers In Computing

Chuck Peddle One of computing's founding fathers, this man designed the 6502 chip, the PET and the Sirius 1 180

Seen In Perspective Vector graphics is the means by which some arcade machines produce sharp line drawings in perspective INSIDE BACK COVER

Next Week

- **We take a close look at Texas Instruments' TI99/4A — at less than £100 it's rather like being offered a limousine for the price of a minicar. But even limousines have snags...**
- **By storing a digitised 'picture' of the building blocks of speech, and then stringing them together at the right speed and rhythm, computers can talk back to us**
- **We enter the realm of sci-fi and look at computer animation: is it just an illustrator's aid or a whole new art form?**

YOUR BINDER ORDER FORM IS WITH THIS ISSUE.

Binders may be subject to import duty and/or local tax.

Overseas readers: This binder offer applies to readers in the UK, Eire and Australia only.

Editor Richard Pawson; **Consultant Editor** Gareth Jefferson; **Art Editor** David Whelan; **Production Editor** Catherine Cardwell; **Staff Writer** Roger Ford; **Picture Editor** Claudia Zeff; **Designer** Hazel Bennington; **Art Assistants** Steve Broadhurst, Liz Dixon; **Sub Editor** Tracy Ebbetts; **Researcher** Melanie Davis; **Contributors** Tim Heath, Richard King, Henry Budgett, Bob Chappell, Brian Morris; **Group Art Director** Perry Neville; **Managing Director** Stephen England; **Consultant** David Tebbutt; **Published by Orbis Publishing Ltd: Editorial Director** Brian Innes; **Project Development** Peter Brookesmith; **Executive Editor** Chris Cooper; **Production Co-ordinator** Ian Paton; **Circulation Director** David Breed; **Marketing Director** Michael Joyce; **Designed and produced by Bunch Partworks Ltd; Editorial Office** 39 Goodge Street, London W1; **© 1983 by Orbis Publishing Ltd: Typeset by Universe; Reproduction by Mullis Morgan Ltd; Printed in Great Britain by Artisan Press Ltd, Leicester**

HOME COMPUTER COURSE – Price UK 80p IR £1.00 AUS $1.95 NZ $2.25 SA R1.95 SINGAPORE $4.50 USA and CANADA $1.95
How to obtain your copies of HOME COMPUTER COURSE – Copies are obtainable by placing a regular order at your newsagent.
Back Numbers UK and Eire – Back numbers are obtainable from your newsagent or from HOME COMPUTER COURSE. Back numbers, Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT at cover price. AUSTRALIA: Back numbers are obtainable from HOME COMPUTER COURSE. Back numbers, Gordon & Gotch (Aus) Ltd, 114 William Street, PO Box 767G Melbourne, Vic 3001. SOUTH AFRICA, NEW ZEALAND, EUROPE & MALTA: Back numbers are available at cover price from your newsagent. In case of difficulty write to the address in your country given for binders. South African readers should add sales tax.
How to obtain binders for HOME COMPUTER COURSE – UK and Eire: Please send £3.95 per binder if you do not wish to take advantage of our special offer detailed in Issues 4, 5 and 6. EUROPE: Write with remittance of £5.00 per binder (incl. p&p) payable to Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT. MALTA: Binders are obtainable through your local newsagent price £3.95. In case of difficulty write to HOME COMPUTER COURSE BINDERS, Miller (Malta) Ltd, M.A. Vassalli Street, Valletta, Malta. AUSTRALIA: For details of how to obtain your binders see inserts in early issues or write to HOME COMPUTER COURSE BINDERS, First Post Pty Ltd, 23 Chandos Street, St. Leonards, NSW 2065. The binders supplied are those illustrated in the magazine. NEW ZEALAND: Binders are available through your local newsagent or from HOME COMPUTER COURSE BINDERS, Gordon & Gotch (NZ) Ltd, PO Box 1595. Wellington. SOUTH AFRICA: Binders are available through any branch of Central Newsagency. In case of difficulty write to HOME COMPUTER COURSE BINDERS, Intermag, PO Box 57394, Springfield 2137
Note – Binders and back numbers are obtainable subject to availability of stocks. Whilst every attempt is made to keep the price of the issues and binders constant, the publishers reserve the right to increase the stated prices at any time when circumstances dictate. Binders depicted in this publication are those produced for the UK market only and may not necessarily be identical to binders produced for sale outside the UK. Binders and issues may be subject to import duty and/or local taxes, which are not included in the above prices unless stated.

Adventure Playground

The computer can be a source of constant entertainment. In an Adventure game you are not just an onlooker but a participant

Mention the word Adventure and most people think of a book, a film, a television programme, perhaps even a personal experience. But many people would think of computers, because there is an army of computer owners to whom the word Adventure (with a capital A) refers to a very specific kind of computer game.

To get an idea of what computer Adventures are like. compare them with books. When you read an adventure story you enjoy the dangers, mysteries and exciting events — but they are happening to someone else. In a computer Adventure, you do not sit observing the action — you are part of it. As the leading protagonist in the story, you are plunged into the action and it is you who lives out the experience.

In a book, the reader cannot influence the course of events of the story. The order and outcome of happenings is always the same and no amount of re-reading will alter it. In a computer Adventure, your decisions, judgments and actions determine how the plot unfolds. There can be any number of variations to the order of events, and many different endings, some pleasant, some unpleasant. The essential thing about computer Adventures is that you take an active part — though in the comfort and safety of your own home.

Each Adventure takes place in a particular environment. This might be a strange world under the earth, a fairground, a ghost town, another planet, a mythical land — in fact, just about anywhere. It can be set in the past, present or future.

The Adventure usually has a consistent theme that includes an ultimate objective. For example, you might have to escape from an alien planet, find and destroy an evil wizard, rescue a princess, collect treasures, or solve a crime and arrest the perpetrator.

Interesting as these factors are, the pleasure in playing an Adventure comes in solving the puzzles. These form an intrinsic part of the Adventure. A puzzle may be encountered in one of four situations. The most common situation is one where the puzzle must be solved before further progress can be made — such as when confronted with a dangerously dilapidated bridge. At the other extreme, the puzzle constitutes a red herring — having crossed a canyon to tackle the strange figure who has been staring at you from the other side, you find only an enormous mirror. The puzzle might be one to which the solution is helpful but not essential for the successful completion of the game — the discovery of a secret passage leading past a vicious Troll, perhaps. It could be a matter of life and death — you are stranded on a ledge with no obvious way up or down, and no food or drink.

Puzzles are capable of being solved by common sense and require no special expertise or knowledge. However, the Adventurer must be on the alert as clues to the solutions are always to be found in the text, or deduced from it. Random elements, except in very small doses, have no place in the well written Adventure.

IAN DOBBIE

It is likely when playing an Adventure that you will come across objects, messages and characters that seem to have no relevance to the story. Bear in mind at all times that almost everything in an Adventure has a specific purpose, even if occasionally that purpose is to throw you off the track. Of what significance is a collection of broken rum bottles? What notice should one take of a hollow voice saying, 'PLUGH'? How can one make practical use of a pile of evil-smelling mud? Why was this rug nailed to the floor when there was nothing under it? These puzzles all come from actual Adventures and were essential components of the plot. When you first find an object, no matter how mundane or strange, you are unlikely

Leading Role
Adventure games such as Dungeons and Dragons have existed for many years, but the home computer has recently introduced them to a far wider range of players. Adventure games are starting to rival arcade-style games in popularity. Though most can still be played only by one person at a time, they can be an absorbing recreation for all members of the family. The player takes on the role of an historical or fictitious character, and travels widely in search of treasure or some other goal

to be certain of its immediate relevance — but as sure as dragons' eggs are dragons' eggs, you will need it before you are done.

Many Adventures have a small maze in which every room or location is described in identical terms. The only sure way to map your way through such labyrinths is to emulate Hansel and Gretel and lay a trail by dropping objects, thereby uniquely identifying each room. This method has become so well-known that some authors have added extra problems — such as having a thief following you in to discreetly rearrange the objects behind you.

In some Adventures, although you must solve every puzzle and achieve every objective to complete the game successfully, the order in which the mysteries are cleared and the goals are reached is unimportant. This is in contrast to those Adventures where there is only one path to a triumphant conclusion.

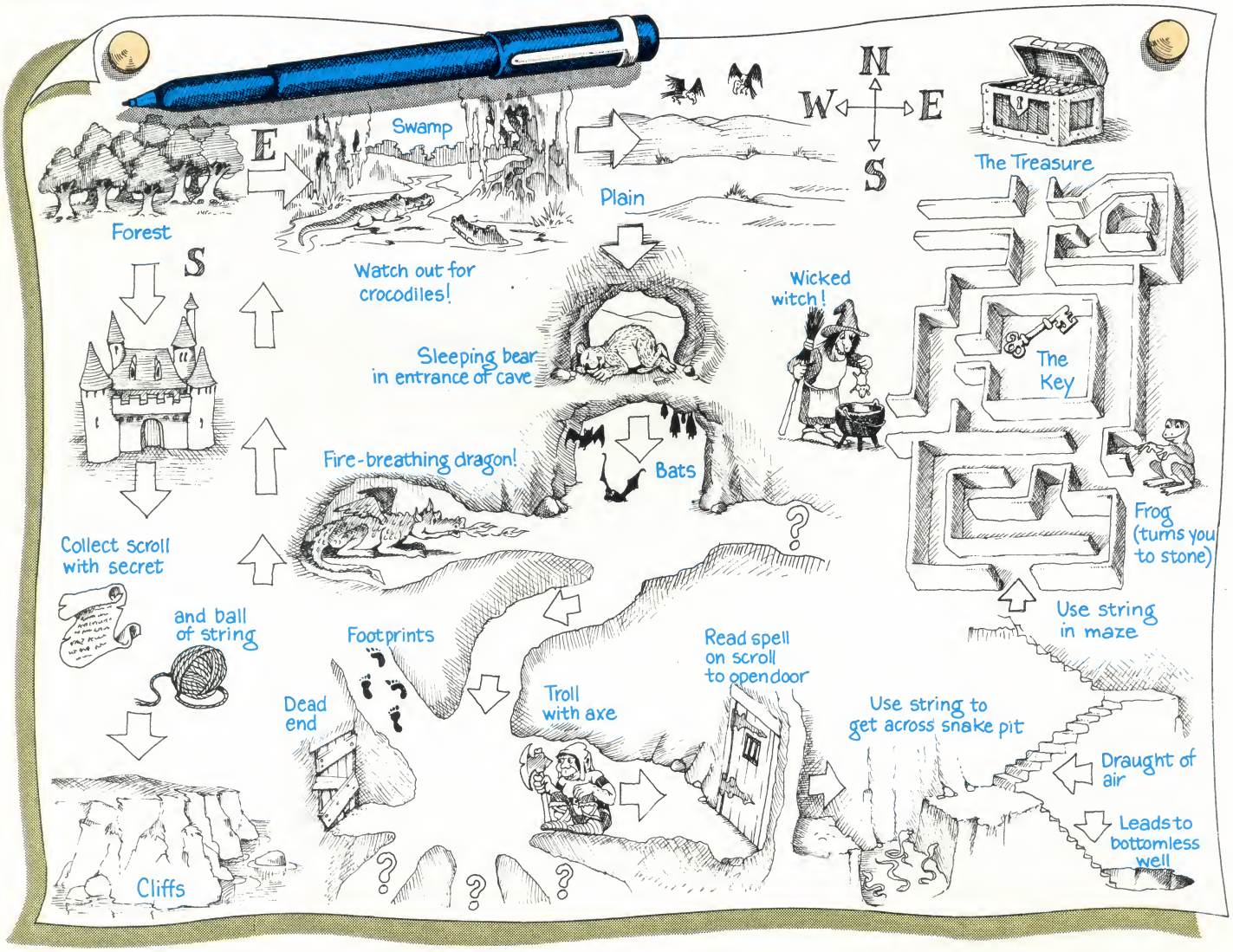
A good Adventure may take you hours, even weeks of playing before it yields its secrets. It must allow you to save the state of the game at any time onto cassette or disk, so that you can resume play later. This is also useful when you reach a dangerous part of the Adventure — striding boldly towards an army of Orcs armed with only a lamp and a bottle of water is asking for trouble. The prudent Adventurer saves the state of the game before striking up a conversation with the Orcs. Then, if the Orcs feel like having the Adventurer for supper, at least the game can be restarted and the Adventurer can try a different course of action — far less vexing than beginning the Adventure again from square one.

Even if the Adventurer is killed, that may not be the end of the journey. Some authors permit the deceased to be brought back to life, often in a puff of orange smoke, sometimes with a loss of possessions and/or points, and always in a less than desirable location, such as the nether-world, or the middle of nowhere.

How do you take part in and communicate with the program? You can be addressed directly or represented by a 'puppet', a character whom you control with commands. The computer acts both as interpreter of your wishes and as narrator. The player enters commands through the keyboard and the computer's responses are displayed on the screen.

Some Adventures display only text on the computer's screen, some purely graphics, others a

Map Reading
In an Adventure game, the player or the character he is representing has to move within some vast terrain or environment — a network of underground passageways is common. Some manufacturers supply hints in the instruction manuals for those who get completely stuck, but the only way to discover the layout is to work your way through it. Such games can take several weeks to complete, so keeping a rough sketch of where you've been and the obstacles you've encountered is essential



DAVID HIGHAM

mixture of the two. Sound effects are used mostly in graphics-only Adventures. Text-only Adventures can be thought of as un-illustrated books, the words describing places, objects and events. The graphics in most text-and-graphics games serve to supplement the textual descriptions and are often static scenes of locations and objects. They range from simple line-drawings to detailed pictures. The graphics in graphics-centered Adventures are normally used for stylised maps or pictures of terrain, and for representing the interiors of buildings. Characters and objects are represented by symbols or miniature figures. In these, the player is usually restricted to a small number of commands, mostly single keystrokes for moving and controlling the character.

The textual display in an Adventure generally covers three elements: where you are, what you can see and where you can go. For example, the screen text might read: 'You are in a dark forest. Above you, the sky is blotted out by dense foliage. There is a well-worn track leading east and west. Just ahead, to the north, a yawning pit beckons. You can see a sword lying on the ground; around the sword is curled a green snake.' You are given a description of your immediate surroundings, some of the directions you might take, and the objects in view.

Commands usually consist of two words, a verb followed by a noun, although the more sophisticated Adventures understand full sentences. Some of the standard verbs include GET, DROP, PUSH, PULL, THROW, LIGHT, KILL, EAT and DRINK. GO NORTH, for example, would be the usual way of specifying movement, although most Adventures allow you to abbreviate these — for example W for GO WEST.

EXAMINE is an essential verb — it is often the means of acquiring further information. EXAMINE SNAKE in the illustration (left) might result in the message 'It is a grass snake', or perhaps 'The snake notices your movement towards it and strikes at you'. Some verbs do not need a noun.

INVENTORY is used to tell you what you are carrying. Some objects can be placed inside others — water in a bottle or an axe in a bag for example — while some may be worn — a ring, perhaps, or a cloak.

SCORE is often used to let the player know what progress towards the objectives has been made. Typing HELP may result in a hint for overcoming a difficulty, but usually advises you to keep trying. Occasionally, in response to a particular command, the Adventurer may be told 'You can't do that — yet!', which implies that the verb-noun combination will give a result but not now, and perhaps not in this place. Part of the challenge comes in finding out which verbs and nouns are relevant to the Adventure. Words and combinations that the Adventure does not recognise are often met with a response such as 'I don't understand you.'

Most Adventure publishers supply hint sheets for those who get stuck. These offer subtle clues.

Some Adventures are too large for the computer's memory. To overcome this, they are supplied on disk, the main program being loaded into memory at the start and chunks of text or graphics swapped in and out from the disk as needed. But thanks to clever techniques for compressing large amounts of text into restricted memory, and the use of machine code programming, a large Adventure can now be stored in a computer's memory, which means that such Adventures can be supplied on cassette tape.

There are Adventures for virtually every micro. Any one of them should give you an enjoyable experience and could be the start of a lifetime hobby.

Happy adventuring!

IAN McKINNELL

Treasure Hunting

The Hobbit is a graphic Adventure for the Sinclair Spectrum and takes its name from the story by J. R. R. Tolkien, a copy of which is supplied with the cassette. You take the part of Bilbo, journeying through Middle Earth, and encountering many of the other characters and events from the book, in search of the dragon and his treasure

COURTESY OF CENTRESOFT LTD.

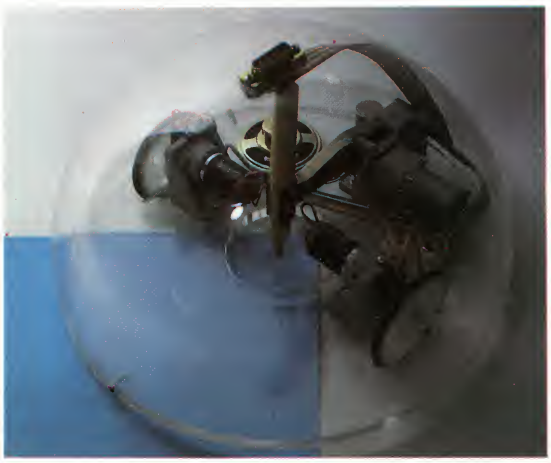
IAN DOBBIE

On The Case

Deadline is a variation on the Adventure theme. You play a detective who has to solve a murder. Your file contains such items as the results of the post mortem, some pills found near the body, and other case notes. The game is not played in real time (a couple of weeks is considered fast), but every action you take, such as moving from room to room, or interviewing a suspect, knocks vital minutes off your 12 hour limit

Microworlds

Computers are widely used in schools, but most educational programs are merely electronic textbooks. Logo is different — it uses a computer to provide a new kind of 'learning environment'



Ian McKINNELL

Walking Turtle
The turtle was designed as a device to think with — particularly in the course of learning about geometry and spatial relationships. When children are unsure how to instruct the turtle to perform a particular manoeuvre, they tend to act out the role of the turtle by walking around the floor and obeying the LOGO instructions. This makes learning a far more 'real' experience

Since the micro first appeared in 1977, far-sighted educators have been quick to identify its potential as a teaching aid in schools. Most schools now have at least one machine, and many offer computer studies as part of the curriculum. Nevertheless, the micro has made few inroads into traditional teaching methods.

The strongest evidence for this lies in the range of educational programs currently available for home computers, which in the main exhibit a remarkable lack of imagination. The majority can be described as 'electronic textbooks', in which the computer presents the pupil with a series of 'frames' on the screen (equivalent to pages in a textbook) and then tests how well the information has been absorbed by means of a series of multiple-choice questions, which the computer marks automatically without the need of a teacher.

Such packages are easy to write on a home computer and offer the benefits of colourful — perhaps even animated — graphics as an accompaniment to the text. This is, however, merely automating the existing process rather than applying the power of the microcomputer in new ways.

LOGO is different. Primarily the work of Professor Seymour Papert, of the Massachusetts Institute of Technology, LOGO is defined as being 'a philosophy of education, and a family of computer programming languages designed to help implement that philosophy'.

Many people have mistakenly viewed LOGO purely as a programming language and compared its commands and facilities with those of BASIC, concluding that LOGO is a far better language for beginners. This misses the point. Papert never intended his system to be a method of teaching children to learn programming. He conceived it as an environment in which children could learn diverse subjects — one in which they could, in fact, learn how to learn.

Much of this philosophy derives from the eminent Swiss educational philosopher Jean Piaget, who argued that, given the right environment, children can learn any subject for themselves in the way that they learn to walk and talk. Piaget's work, however, was entirely theoretical, and Papert set out to produce a practical environment for Piagetian learning.

That traditional education methods don't achieve this is evident from the fact that the majority of adults are afraid of learning and do not enjoy the idea of having to acquire new skills or areas of knowledge. The most common symptom of this, argues Papert in his book *Mindstorms — Children, Computers, and Powerful Ideas,* is the widespread fear of mathematics — or 'mathophobia', as he calls it.

One of the reasons for this is that most subjects are taught in the same way, whereas their applications are completely different. Children are taught, for example, to multiply in the same way they are taught the capital cities of the world — by rote learning. The learning process has been divorced from what is being learned, when it should be inseparable.

Papert himself views learning a new skill, whether it's flying, cooking or a foreign language, as a hobby. He attributes this attitude to his childhood, when he discovered how gearwheels work at an early age and applied this concept whenever he came across a new problem. Albert Einstein, too, used to say that when he encountered something he didn't understand, he would break it down into concepts that he had grasped before the age of five.

These powerful notions were incorporated into LOGO, as can be seen in our example of LOGO in action. The first important feature of LOGO is the turtle, which was designed as 'a device to think with' in the same way that Papert used gears as a child. For young children, the turtle takes the form of a specially designed floor robot (see page 176), which is linked to a micro and can be moved around the floor by typing in commands in LOGO. The turtle usually carries a pen for marking shapes on the floor, and it can also feature a small loudspeaker and collision detectors that guide it through an obstacle course.

Children usually graduate from using floor turtles to screen turtles — shapes that can be moved around the computer's screen. The turtle is

Learning Curve

This fictitious but typical example shows how LOGO encourages a group of children to solve problems they haven't encountered before.

```
TO CURVE
REPEAT 80
    FORWARD 1
    RIGHT 1
END
CURVE
```

'If we want petals then we need to draw a curve.'
'But the turtle always moves in straight lines.'
'What if we made it move a short distance, turn just a little bit, then move forward again, and so on — that would be like a curve.'
'OK, the smallest distance is one, and the smallest angle is one. Let's try doing that eighty times.'

```
CURVE
CURVE
```

'There, that's just what we want.'
'Two of them will make a petal — let's try it.'

```
TO PETAL
CURVE
RIGHT 90
CURVE
END
PETAL
```

'That's nothing like a petal — what happened?'
'It just carried on from the last curve — we should have told it to go in another direction.'
'But how much do we make it turn?'
'Let's try ninety — that often works.'
'And let's make a new word — PETAL — that will save time.'

```
TO PETAL
CURVE
RIGHT 100
CURVE
END
PETAL
```

'That's better, but ninety wasn't enough — what shall we try next?'
'Let's try and work it out, instead of guessing.'
'Yes — remember we learnt that if the turtle goes right round anything it turns through a total of three hundred and sixty.'
'Well, we know it turns through eighty on the first curve, so it must turn through eighty on the way back — that makes one hundred and sixty.'
'Leaving two hundred to be turned at the point of the petal.'
'No, because to get back to the position it started from, it would need to turn round at the other end too.'
'So we should try half of two hundred.'

```
PETAL
PETAL
PETAL
PETAL
```

'Great — four of those will make a flower.'

```
TO FLOWER
REPEAT 4
    PETAL
    RIGHT 10
END
FLOWER
```

'That's not much good, it's !op-sided.'
'We forgot to put in any turns between the petals.'
'But why didn't it draw them on top of each other, then?'
'Because when it's drawn one petal, the turtle is facing one hundred to the left of where it originally started.'
'So the petals are turned round by one hundred to the left each time.'
'That looks about right — what we want is ninety, so let's add in a right turn of ten between each petal.'

```
FLOWER
RIGHT 180
FORWARD 100
RIGHT 180
PETAL
```

'At last! Turn the turtle round and we can draw the stalk.'
'One hundred ought to be long enough.'
'Let's have a leaf on the end — it can be the same as a petal.'
'But remember the angle this time — the turtle needs to be turned right round.'

LIZ DIXON

a powerful device with which children can learn the basic concepts of spatial relationships, taking them up to advanced geometry.

Control of the turtle, however, is only one small application of LOGO, but because it is the most visually interesting it is the most publicised aspect. Of greater importance is the concept of building up simple ideas into more sophisticated ones, and conversely the breaking down of large problems into smaller problems of a kind that have previously been tackled.

These processes can be clearly seen in the imaginary conversation of a group of children learning to instruct the turtle to draw a flower (see box). They start off with only three available commands: FORWARD — which moves the turtle forward by a specified amount; RIGHT — which turns the turtle through a specified angle; and REPEAT, which repeats the lines indented in the program a specified number of times.

From these fundamental ideas the children first construct a 'tool' — a program — for drawing a curve (TO CURVE . . . END). This whole sequence can now be called up simply by typing CURVE. Similarly, after experimentation and further learning, a PETAL command is defined, which makes use of the CURVE command. Eventually a command FLOWER, which will draw the complete picture, is developed.

LOGO is not the only language to incorporate such structures (another is FORTH — see page 150), but it is the only one designed to be used by young children. It does away with many of the formalities and procedures associated with programming in other languages. Indeed, the aim was that the child shouldn't be aware that he is programming a computer — only that he is solving a problem.

In some learning situations, the pupil does not even get involved at this level. The teacher sets up a series of powerful tools using LOGO, which all relate to a particular subject or area of knowledge. The child is then allowed to explore the subject using the tools and discover it for himself. These areas are called 'microworlds' — limited environments in which the computer is used to simulate something in the real world or some area of knowledge.

The best example of a microworld is probably the LOGO model of Newtonian physics. Though Newton's First Law states that without the influence of external forces a body will continue to move in a straight line at a constant speed, young minds observe that in the real world everything slows down. This causes a blockage to learning. Using LOGO, however, a microworld can be set up in which everything behaves in true Newtonian fashion, and with the aid of tools to push objects around the screen, children soon learn all three of Newton's laws for themselves.

LOGO is a powerful concept well worth learning about with a home computer. Floor turtles are available but are not cheap. Versions of LOGO that use screen turtles are becoming available for several popular home computers.

The Age Of Portables

As computers become more sophisticated they can be squeezed into smaller packages, so that truly portable micros are now possible

IAN DOBBIE

No Executive Toy
Portable computers such as the Epson HX-20, with up to 32 Kbytes of memory, built-in peripherals and a wide range of software, have greatly increased the amount of information available to a business executive, no matter where he may be. Each microcassette can hold up to 230 Kbytes (perhaps 40,000 words, equivalent to a powerful database), while standard word processing and spreadsheet software allows work to be done almost anywhere — even in a taxi cab!

Development of the portable micro as we know it today has come from two directions. One has been the augmentation of the pocket calculator, as in the Sharp PC1251 and the Casio FX700P. The other has been an evolutionary process of miniaturisation, which has resulted in such machines as the Tandy TRS80 Model 100 Personal Computer and the Epson HX-20 (see page 169).

These advances were a result of the development of more densely packed chips, allowing a great deal more information to be contained in the same physical space.

With the advent of the single-chip microprocessor in 1972 it became theoretically possible to build an entire computer into a box no bigger than a cigarette packet. However, the size of the display and the prime means of access — the keyboard — imposed practical limitations on this miniaturisation.

Pocket calculators did indeed become smaller, and today digital watches are available that double as calculators, requiring use of a matchstick or a purpose-built stylus to operate the 'keys'. But it is difficult enough to perform a simple arithmetical operation on these and even the most enthusiastic would shrink from entering a 50-line program in BASIC. So personal computers are unlikely to become as small as a wristwatch. But they have become calculator-sized.

Once pocket calculators became programmable, albeit in their own programming notation, it was only a short step to the incorporation of a high level programming language, and the obvious choice was BASIC. At about the same time manufacturers started to use 'non-volatile' RAM — a type of memory that, by retaining a small electrical charge, doesn't lose its contents when the power is turned off — and a more comprehensive character generator, to allow for alphabetic characters to be displayed in addition to numerals.

Reasonably priced devices such as the Sharp and Casio pocket computers, which have a range of BASIC commands comparable to home computers and similar memory capacities, are fast taking over the programmable calculator market. Packaged to fit comfortably into a pocket — even a shirt pocket — some offer the possibility of being directly interfaced to both a printer and a cassette recorder. Doubtless, a built-in communications facility will follow if there is sufficient demand, allowing the transfer of information over telephone lines. The Casio range, especially, is attractive to scientists and engineers, as it retains the wide range of mathematical and scientific functions that made that company's calculators desirable.

Other models are available with a built-in printer and, complete with cassette interface, cost around £125. Sharp's PC1251 is similarly priced for a similar specification.

The next major step is represented by the 'lap-held' models such as Epson's HX-20, the Tandy 100 and the NEC 8220. These micros offer a full implementation of BASIC (in all cases Microsoft), 16 to 64K bytes of usable RAM, an inbuilt liquid crystal display (LCD) of a reasonable size (20 × 4 characters on the Epson, 40 × 8 on the other two) and the ability to connect to a wide range of industry-standard peripherals.

Indeed, couple them to a regular monitor (you may need to buy a special interface) and you have a micro comparable in capacity and performance to a conventional home computer at the same reasonable price. The essential difference lies in the fact that these machines rely on their own internal power supply, thus allowing absolute portability.

The Epson, for example, has built-in nickel

cadmium cells, one full charge of which lasts up to 50 hours, and the system is designed so that when the cells are running low it shuts itself down to preserve essential data.

In contrast, the NEC and Tandy use complementary metal oxide semiconductor (CMOS) circuitry, which requires much less power and enables them to run on ordinary pen-light cells.

The Epson, subject of the first mass-appeal advertising campaign in the marketing of micros, was originally seen as an 'executive aid', but it is more likely to be used outside the office than inside it. It is perfectly suited to data capture (the gathering of information) on the factory floor, on remote sites or even while walking down the street — to be processed back in the office.

In practice the Epson offers various ways of transmitting data. Proprietary add-ons allow it to be used as a Telex terminal; through a modem it can be connected to another machine via the telephone lines; or at the cheapest and simplest, data can be stored on cassette tape and sent through the post.

It is just this flexibility and versatility that makes these machines so attractive to people who had not previously considered that computers could be used in their everyday work. A salesman, for example, making 20 or more visits a day simply to record regular orders, might use a portable machine of this capacity and power rather than fill in order forms. The program to process orders would 'prompt' him with a product list and when the customer placed an order he would log the quantity. At the end of each call the order data would be dumped onto tape, and at the end of the day the salesman could post the tape to head office or transmit the information directly into the company's main computer system.

The micro's own printer would provide the salesman with a copy of his day's orders and his customer with an immediate confirmation.

Slightly more sophisticated versions might also refer to stock levels and give a warning when these fall dangerously low, though where more than one salesman was employed the micro would have to be connected to the mainframe at each call. This would present no problems, given a cheap, lightweight acoustic coupler. At this point our cheap portable turns into an interactive terminal, allowing the whole of the company's database to be interrogated with an immediate response.

Apart from the obvious benefits of immediate access to up-to-date information, the savings in time and cost brought about by this sort of on-line data entry can pay for a complete portable system in a matter of months.

Just as smaller pocket computers are ousting programmable calculators, lap-held machines are taking over from hand-held data-logging/data-collecting devices. These battery-powered 'dumb terminals' — which cannot be programmed — have been available for some time, but they have never been popular or successful due to high cost and difficulty of operation.

Given suitable software, the ability to plug in to office-quality monitors and printers, and perhaps disk or cassette tape drives with faster access, lap-held portable computers are likely to become common. Portable machines of this sort provide the first real market for bubble memory — a self-contained, non-volatile memory based on a radically different type of chip, which offers huge storage capacity in a small package. Typical applications, such as Sharp's PC5000, provide 128 Kbytes of storage per plug-in cartridge, with an access time rather quicker than disk. The prospect of small machines with a million bytes of memory incorporated is very attractive.

This storage capacity is currently available on disk in the third type of portable machines such as the Osborne Executive, the Ajile Hyperion and the Portico Miracle. Costing up to five or six times as much as the HX-20 or TRS100, these machines are better described as transportable, because they need mains power. Battery packs are available for some machines, but these rarely offer more than a couple of hours' use.

Their specification usually includes twin floppy disk drives, built-in television monitor, detachable

A Machine For All Seasons
(Clockwise, from top right.) Epson's HX-20, first of the new breed of 'lap-held' self-contained computers, offers up to 32 Kbytes of RAM and a wide range of peripherals.

Least powerful — and least expensive — among the available range of portable computers are the Sharp PC1251, shown here with printer and microcassette drive, and the Casio FX700P which uses a standard domestic cassette recorder.

After capturing a large share of the market for 'transportable' computers, the Osborne Computer Company's second model, the Executive, offered a larger screen and other refinements

IAN McKINNELL

IAN McKINNELL

Joined-up Writing
The concept of an 'electronic notebook' has come one step closer with the introduction of the Microwriter, a purpose-built word processor suitable for one-handed operation. The six keys are pressed in different combinations to achieve the full alphabet. Output can be either direct to a printer, with some formatting commands to allow you to arrange the text, or into a desktop word processor for storage

Tools Of The Trade
Inexpensive portable computers have found uses in areas previously inaccessible to information processing systems

Journalists can enter into their portable machine the questions they wish to ask at interviews, enter the answers alongside, and then edit the piece into shape on the train, perhaps, on the way home.

Pharmacists have a cheap and easy way of labelling prescriptions legibly that also eases the task of controlling their stock.

Salesmen can give estimates on the spot even for complicated calculations like central heating installation or life insurance

Stock control is very important even in a small business, but in industry, where stock values run into many millions of pounds, it is vital to keep an accurate record. Portable micros, with 32 or 64 Kbytes of RAM, back-up storage on cassette tape, and communications facilities, are ideal for this purpose

DAVID HIGHAM

keyboard and an operating system such as CP/M.

First on the market was Adam Osborne's model 1, which offered a complete range of software, including BASIC, Wordstar, and Supercalc and CP/M utilities, for less than £2,000.

Its weak point was a small screen — 100 × 85mm (4 × 3½ins) — which made the characters rather small and demanded the use of horizontal scrolling to achieve a readable display size when a word processing package such as Wordstar was used. Users, however, soon accustomed themselves to it and the Osborne 1 became a market leader.

Competitors came to offer more facilities, especially in terms of microprocessor power and speed, and capacity of disk storage, until this class of machine became indistinguishable in all but looks from sophisticated office/business systems. In many cases they offer full compatibility with specific machines, notably the IBM Personal Computer, which perhaps encourages their use as a 'second computer'.

The real difference between these portables and office systems lies in their packaging. An applications program that runs on a normal office system will run on a portable with a similar microprocessor. The only limitation is still likely to be the size of the monitor screen and hence of the individual characters.

All this has been made possible by the development of the single chip microprocessor. The limitations on input and output accessibility are a reflection on our dependence on the keyboard as the main input device and the cathode ray tube as the main output device. But what if we suggest voice recognition and speech synthesis as an alternative? There is then no reason why a micro of similar power to the Ajile Hyperion, for instance, should take up any more space than a Sony Walkman — and the chances are that, had we the technology available, it would look just like that, with the addition of a miniature microphone.

In the present climate of high-speed technological advance the developments in this field are likely to be considerable, even if we don't stray into the fantasy world of speech input and output. Imagine, for example, combining the Sinclair ZX Spectrum with Sinclair's flat-screen television and Microdrive, and powering the system from a nickel cadmium rechargeable pack: there is no reason why one should not put this together for oneself — though perhaps Sinclair will do it for us...

Because of its small size, the ZX Spectrum is an obvious machine for this application, but most computers operate on the sort of voltage obtainable from standard dry cells: there is little reason why almost any computer could not be made portable in this way.

An interesting trend is typified by the ACT Apricot, which is a normal office system, though small in size. Leave aside the monitor and the Apricot becomes a portable system. 'Portability' is, of course, a vague term. Many micros that are so described are merely fitted with a carrying handle and were never conceived as purpose-built, self-contained units.

With the introduction of a wide range of portable micros, the industry has taken a huge step towards fulfilling its real potential — making computing power available to all at low cost and in a form requiring no prior knowledge or training. Admittedly, it will be some time before these criteria are properly met, but the goal is in sight.

Epson HX-20

Not only can this 'lap-held' machine be carried anywhere — it can be used for your office work while you're travelling

The Epson HX-20 was the first genuinely portable computer. As it is fully programmable in BASIC it makes possible far more applications than the most sophisticated calculators. Yet a self-contained design and total weight of less than two kilograms means that it can be carried in a briefcase.

Much larger machines that had hitherto been sold as 'portables' were rapidly christened 'luggable' by the trade. The HX-20 has sold across the whole spectrum of microcomputer users: home enthusiasts, businessmen and engineers.

Because the HX-20 is not based on a standard design but is completely new, there is still relatively little software available. However, it is an ideal machine for learning to program BASIC. Often, the purchaser's intention is to write a program that will perform a specific (perhaps unusual) application: an estimation program for insurance salesmen, a navigational aid for yachtsmen, and a note-taker for journalists — all have been devised for the Epson.

The unit comes with a liquid crystal display that can handle up to four lines of 20 characters, or simple graphics up to a resolution of 120 x 32. In most applications, this display acts as a 'window' that can be moved by the cursor keys (marked with arrows) to show any section of a much larger text area handled by the computer.

The built-in printer mechanism uses ordinary paper on a five centimetre (two inch) wide roll, on which it can print up to 24 columns of text.

The microcassette drive is an optional extra but is usually shown as if built in because most people require it. The space it occupies can be used for solid-state cartridge software, though none has yet been produced. The microcassette is superior to a domestic cassette recorder. The computer can sense the position of the tape, and it moves automatically 'fast forward' to find the required program or piece of data.

The range of interfaces on the back and side of the casing reflects the diversity of intended applications; there is even a socket for a bar code reader pen (see page 40). The 16 Kbytes of RAM can be expanded to 32 Kbytes, using an add-on pack on the side.

Inexpensive hardware and software are available to enable the HX-20 to communicate over the telephone line — either to a like machine or to a mainframe computer to gain access to central information.

CHRIS STEVENS

Epson HX-20 Keyboard
It is the full typewriter-sized keyboard that really determines the dimensions of the HX-20. Though the 'feel' of the keys is somewhat different from a normal typewriter, it is well-liked by touch typists.

In addition to the numerals along the top row of keys, pressing 'NUM' converts the keys U, I, O, J, K, L, M into a numerical keypad. This is faster for entering large amounts of numerical data.

Cursor movement and other editing keys are found at the top right, together with a SCRN key for scrolling the small screen up and down.

The five programmable function keys (PF1 to PF5) are physically different from the other keys and two of them double up to control the microcassette, and to copy the contents of the screen on to the printer

In-House Printing
Small dot-matrix or type-roller printers have formed an integral part of cash registers — and even calculators — for quite some time, and Epson, who started life as printer manufacturers, were quick to incorporate the idea into their HX-20 portable. This dot-matrix printer is capable of graphics as well as character output

Power Socket
With the transformer connected the HX-20 will draw its power from the mains and recharge its internal batteries.

Reset Button

Bar Code Connector
A reader pen can be connected here for reading the bar codes now found in many shops

Cassette Port
Works with a domestic cassette recorder and includes control over the motor. Performance, however, is not as efficient as with the microcassette

Mass Storage
The HX-20's built-in microcassette recorder, built around the sort of cassette tapes used in pocket dictating machines, is a significant advance over ordinary cassette recorders because the computer controls the 'fast wind' function. The time taken to get to a specified point on the tape is therefore drastically reduced. Another advantage is that the unit plugs straight into a reserved space in the casing — there are no leads to get in the way

Printed Ciruit Board
In our illustration, the PCB has been turned upside-down for clarity. The chips normally face the underneath of the computer, and removable doors in the bottom of the case mean that the important chips can be exchanged without dismantling the entire unit

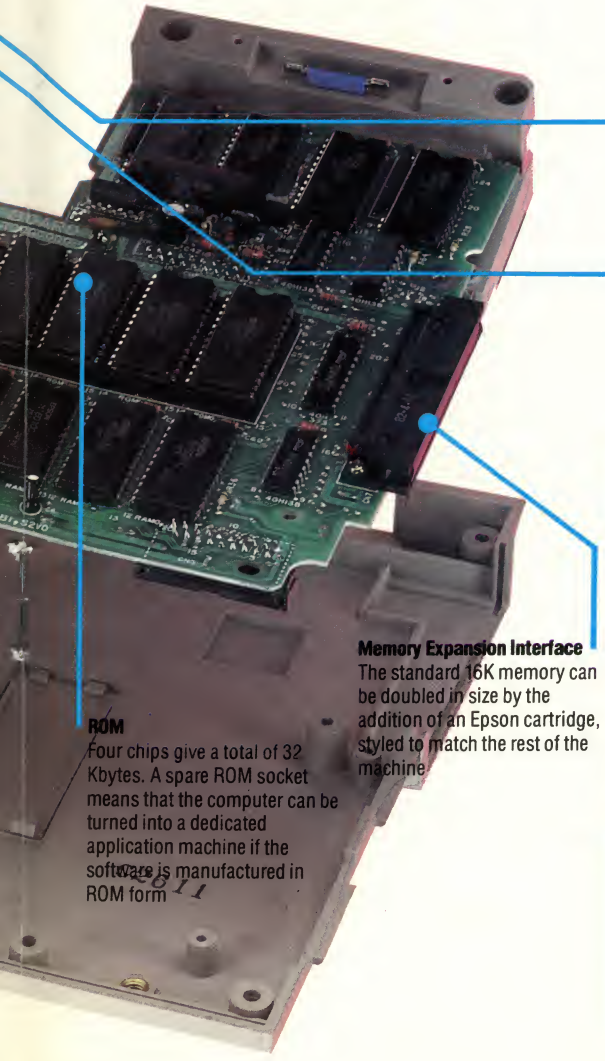
Microprocessors
Two 6301 microprocessors (manufactured by Epson) control the computer and its interfaces. Unusually, each has 4 Kbytes of ROM and 128 bytes of RAM built-in, in addition to the external chips

RAM
The HX-20 comes with 16 Kbytes of RAM as standard, configured as eight 2 Kbyte chips

Cartridge Release
This knob allows the complete microcassette to be removed. In theory this module can be replaced with a solid-state cartridge, if you wish to use the machine for just one purpose

Viewing Angle Adjuster
Liquid crystal displays normally have to be viewed from the correct angle to be legible. Rotating this knob lets the user view from any desired angle

On/Off Power Switch



Serial Interface
Though the RS232 port is also a serial interface, this port is intended to drive various peripherals which Epson will introduce for the HX-20

RS232 Port
This is the accepted standard for serial interfaces and is used to connect to a printer, or to a modem for communicating with another computer over the telephone

Memory Expansion Interface
The standard 16K memory can be doubled in size by the addition of an Epson cartridge, styled to match the rest of the machine

ROM
Four chips give a total of 32 Kbytes. A spare ROM socket means that the computer can be turned into a dedicated application machine if the software is manufactured in ROM form

A New Slant
Much of the physical space taken up in a microcomputer is devoted to the cathode ray tube that forms the heart of the monitor. Epson overcame this problem in the HX-20 by substituting a liquid crystal display for the conventional VDU.

Displays of this type have long been used in calculators, but this application takes a quantum leap in terms of quality as the 4 by 20 matrix is adjustable for viewing angle

EPSON HX-20

PRICE
£411 or
£486 including microcassette

SIZE
290 × 210 × 45mm

WEIGHT
1.7Kg

CPU
6301 Hitachi

CLOCK SPEED
2.45 MHz

MEMORY
16K of RAM, expandable to 32K using a bolt-on pack. 32K of ROM, which can be expanded to 64K by the same means

VIDEO DISPLAY
Liquid crystal display (adjustable for viewing angle) with 4 rows of 20 characters acting as a window onto an area of up to 255 × 255 characters. Every pixel on the display can be addressed, giving a maximum graphics resolution of 120 × 32

INTERFACES
Memory expansion port, cassette, RS232 (for external printer), serial (for future disk), bar code reader pen

LANGUAGE SUPPLIED
BASIC

OTHER LANGUAGES AVAILABLE
FORTH interpreter

COMES WITH
Moulded plastic carrying case, power supply unit, instruction manual

KEYBOARD
60 typewriter-style keys, five programmable function keys, and four dedicated function buttons

DOCUMENTATION
The computer comes with two manuals: an operations manual, and a BASIC reference manual. Both are very comprehensive indeed, but they do suffer from the lack of an index. Epson chose to implement Microsoft BASIC, an industry standard, and the manual for the language is probably good enough to act as a general tutor, as well as a reference guide.

The operations manual includes such specialist information as a complete list of I/O connections for the RS232 and serial interfaces; acceptable signal strength levels; and comprehensive memory mapping tables

Leaving It To Chance

Continuing our look at Basic functions, we come to RND, which produces random (or nearly random) numbers for use in games or statistical programs

Now that we have seen how several of BASIC's functions work we shall look at one of the most commonly used — the RND function. RND is used to generate random numbers. It is also used in games whenever there is an element of chance.

Unfortunately, RND is one of the most variable 'words' in BASIC. Our description of it may differ from the way it is implemented in your home micro. Let's, therefore, clarify the differences between BASIC used in the Basic Programming course and your BASIC.

Most of our programs are based on Microsoft BASIC (or MBASIC). Microsoft is an American company and their BASIC was one of the first made widely available. BASIC is a language with no official standard, but Microsoft's version is as near to a standard as there is. Many other versions are modelled on Microsoft's, and the company has been commissioned to produce versions for several popular computers.

The chief difference between MBASIC and most of the more recent versions is that home computers now have powerful graphics capabilities that were not available when MBASIC was developed. Other versions of BASIC generally include a number of graphics commands and statements. To get the most from your computer, you will want to use its graphics capabilities to the full, and this will require a careful study of the owner's manual.

Of the various BASICS supplied with popular home computers, Sinclair BASIC (used in the ZX81 and Spectrum) and BBC BASIC probably differ most from MBASIC. Texas Instruments' version (used in the TI99/4A) also has a number of significant differences. As far as possible, we point out how to modify our programs in the 'Basic Flavours' boxes and you should refer to these if you have any problems running the programs.

CAROLINE HOLDEN

As mentioned previously, the RND function differs from version to version. Check in your BASIC manual to see how it has been implemented in your version. We are illustrating its use in a very simple dice game. As with previous programs we have done most of the work in subroutines. This technique has the advantage of making the programs more readable, easier to write and easier to debug.

The main program starts with the statement RANDOMIZE in line 20. Most, but not all, versions of BASIC need this statement to 'reseed' the RND function. It is actually quite hard to get computers to produce genuinely random numbers. Without this reseeding operation, the same sequence of supposedly random numbers would be produced each time by the RND function. Line 50 then calls a subroutine that uses RND to assign a random number to the variable D. The form we have used is:

```
320 LET D = INT(10 * RND)
```

This is the line most likely to need changing when you enter the program. Details of how different versions of RND work are given in 'Basic Flavours', so let's see what's happening in this Microsoft BASIC. The RND uses an expression (in brackets, as is usual with functions) as an option to alter slightly the sequence of numbers generated. With no expression — for example LET A = RND — the value of A will be a number between 0 and 1. We do not want a number smaller than 1 so we multiply the number by 10. This can be done like this: LET A = 10 * RND. If, for the sake of argument, RND had returned the value 0.125455, the value of A would now be 1.25455.

To eliminate the fractional part of the number and retain only the integer portion, we use the INT (integer) function like this: LET A = INT(10 * RND). Some versions of BASIC allow the upper limit of the random numbers generated to be specified in the expression used in the brackets after RND. For example, Dragon BASIC will print a whole number in the range 1 to 6 in response to: PRINT RND(6).

Since our Microsoft BASIC cannot do this, we check to see if the numbers returned are greater than 6 or less than 1 as such numbers are of no use in a dice game. This is done in lines 330 and 340:

```
330 IF D > 6 THEN GOTO 320
340 IF D < 1 THEN GOTO 320
```

If D is outside the limits 1 to 6, the GOTOs make the program jump back and try again.

Having chosen a random value for D between 1 and 6, the dice throw subroutine RETURNs to the main program. This prints the message YOUR SCORE IS A, followed by a picture of a dice. Notice how the appropriate picture of a side is selected. It is done in the SELECT subroutine. For example, if the dice (and therefore D) is a 1, line 410 calls the subroutine starting at line 530 thus:

```
410 IF D = 1 THEN GOSUB 530
```

This subroutine is nothing more than a series of PRINT statements designed to produce crude graphics on the screen. Your BASIC may well have much better screen graphics, and if this is the case it would be better to substitute the appropriate graphics statements in place of our subroutines.

Once the program has chosen a dice for you at random, it will then repeat the process to select and display a dice for the computer. The part of the program that decides who has won has been incorporated in the main program; it could just as well have been written as a subroutine, but this would hardly be worth it since it is only four lines long. Line 200 compares M (my dice) with C (computer's dice) to see if they are equal. If they are, the words IT'S A DRAW are assigned to the string variable S$. Line 210 tests to see if M is greater than C. If it is, it assigns the words YOU HAVE WON to S$. Line 220 tests to see if M is less than C. If it is, it assigns the words THE COMPUTER HAS WON to S$. Line 240 simply prints the result and the game is over. Although this program is rather long, it is essentially very simple. It uses only one function, RND, has no loops, no subscripted variables and nothing more complicated than a few IF . . . THEN statements.

Given that the RND function is so variable, and that some versions of BASIC (Microsoft's, for example) require the RANDOMIZE statement to generate a new sequence of random numbers, is there any way we could generate truly random (i.e. unpredictable) numbers without using these functions? Several techniques are available.

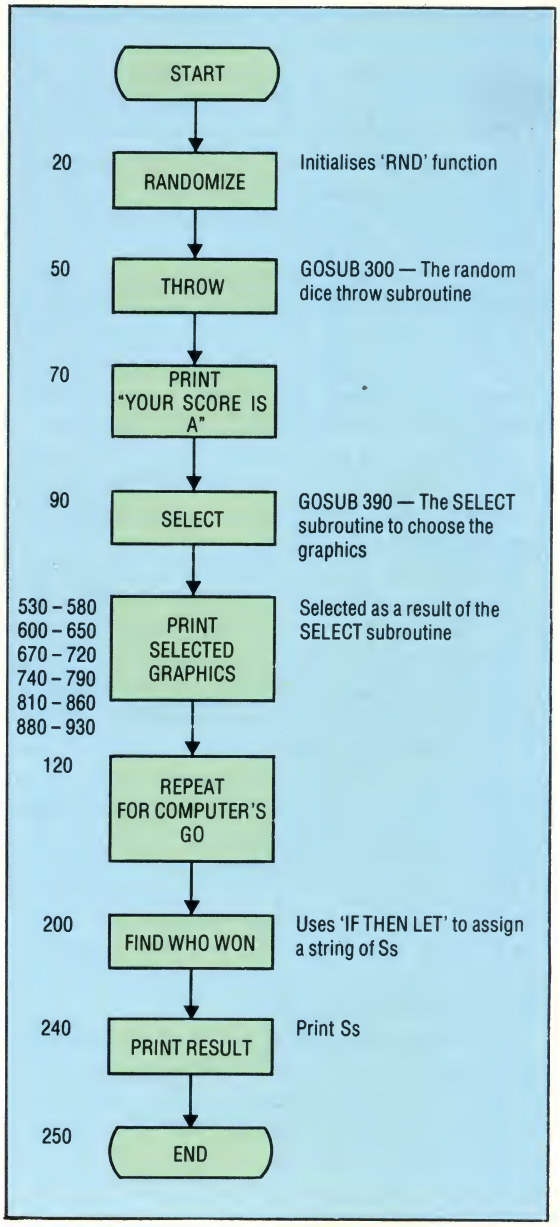
One of the functions we have not looked at so far is INKEY$ (pronounced 'inkey-string'). Each time the word INKEY$ is encountered, the program inspects the keyboard to see if a key has been pressed. The program does not wait for a character to be input as it does when the command INPUT is used. So the command INKEY$ is usually placed in a loop. The program then continually scans the keyboard, waiting for something to be input. There is usually a test within the loop to terminate it, if an appropriate character has been input. This makes it possible to write a program to form a counting loop that will terminate when a specific character has been typed in. What would happen if we used this program?

```
10 PRINT "HIT THE SPACE-BAR"
20 FOR X = 0 TO 1
30 LET R = R + 1
40 LET A$ = INKEY$
50 IF A$ = "  " THEN GOTO 80
60 LET X = 0
70 NEXT X
80 FOR Q = 0 TO 1
90 IF R < 10 THEN GOTO 130
100 LET Q = 0
110 LET R = R / 10
120 NEXT Q
130 PRINT INT (R)
140 END
```

Would R be a random number? It should be, so let's look at the program and see why.

Line 10 prints the prompt HIT THE SPACE-BAR. Before we have time to respond to this prompt, the program has entered the FOR X = 0 TO 1 loop in line 20. 0 and 1 may seem like strange limits for the loop, but we will see how this structure is used shortly. Line 30 assigns the value 1 to variable R the first time through the loop. Line 40 assigns whatever character is typed in on the keyboard to the string variable A$ in line 40. This is done using the INKEY$ function. If you were to hit the letter R, R would be assigned to A$. Line 50 tests A$ to see whether it is a space (this is represented in BASIC as a space between double quote marks thus " "). If A$ is a space, the program branches using the GOTO statement, but if A$ is not a space, the program continues to the next line.

This is line 60, which says LET X = 0. Now X is the index of the loop. The NEXT X statement in line 70 causes the program to return to the beginning of the loop in line 20. Since X has been reset to 0, the loop repeats it. In this way the FOR X = 0 TO 1 loop will be repeated indefinitely, as long as the IF A$ =



Program Flow
The flowchart shows the main actions performed by the program in simplified form. Corresponding line numbers are given on the left and short explanatory notes on the right. This is not a full flowchart as many of the 'decisions' and branches in the program are not shown

" " test fails.

If at some stage the space bar is pressed, A$ will be assigned the character representing a space, and so the program will branch to line 80 and the loop will not be repeated.

But what will happen while the loop is repeating? Line 30 increments the value of R at each repetition of the loop. The first time through, R would be set to 1, the second time through it would be set to 1 + 1 and so on. When the loop has been broken out of by the test on A$, we could read R to see what we had counted up to.

Computers, however, operate very quickly and so R could be in the hundreds by the time we press the space bar. What would we do if we wanted values of R only between 1 and 10? Line 80 sets up another loop to enable us to test R and divide it by 10 if it is larger than 10. As long as R is larger than 10, the test in line 90 will fail, the value of Q will be reset to 0 and the loop will be repeated. Line 110 divides the value of R by 10 and a result is not printed out until the value of R has been reduced to a figure of less than 10. Line 30 ensures that the value of R can never be 0.

In theory, then, this program should produce a random number varying between 1 and 9 inclusive. But does it? The INT statement ensures that the decimal fractions have been removed, therefore the possible values of R would be 1,2,3,4,5,6,7,8,9. The average of these numbers is 5 (because their sum is 45, and 45 ÷ 9 = 5). Try it and see. You could do this by running the program a number of times, noting the value of R each time and then calculating the average. Alternatively, you could add some lines to the program to make it run, say, 100 times, adding the value of R to another variable S and then dividing S by 100.

When we tried this, we found that the average value of R was well below 5 and so the numbers could not have been random. It is instructive to consider why this could be.

The problem is that although BASIC is fast, it is not fast enough. The first loop lets the value of R increment until it reaches hundreds, or even thousands, before we press the space bar. Unless you make a deliberate effort to vary the amount of time elapsing between seeing the HIT THE SPACE-BAR prompt and actually pressing it, chances are you will press it after a fairly regular lapse of time. In this time, the value of R will probably have increased to several hundred.

The divisions that take place to reduce the value of R to a figure below 10 do not take place until after the space bar has been pressed. This means that R will almost always be in the low hundreds before the divisions take place and so the final value of R will tend to be low.

Is it possible to write a routine that overcomes this problem? The answer is yes, if we can make the counting process fast enough for our reaction time to the HIT THE SPACE-BAR prompt to be truly unpredictable. The solution is to make the test for the 'greater than upper limit' part of the first loop. Consider this program:

```
10 REM  GAME OF DICE -- MAIN PROGRAM
20 RANDOMIZE
30 REM  YOUR THROW
40 REM  GOSUB 'THROW' ROUTINE
50 GOSUB 300
60 LET M = D
70 PRINT "YOUR SCORE IS A"
80 REM  GOSUB 'SELECT' ROUTINE
90 GOSUB 390
100 PRINT
110 REM  COMPUTER'S THROW
120 REM  GOSUB 'THROW' ROUTINE
130 GOSUB 300
140 LET C = D
150 PRINT "THE COMPUTER'S SCORE IS A"
160 REM  GOSUB 'SELECT' ROUTINE
170 GOSUB 390
180 PRINT
190 REM  WHO WON?
200 IF M = C THEN LET S$ = "DRAW"
210 IF M > C THEN LET S$ = "YOU WON"
220 IF M < C THEN LET S$ = "COMPUTER WON"
230 REM  PRINT RESULT
240 PRINT S$
250 END
260 REM
270 REM
280 REM
290 REM
300 REM  RANDOM DICE THROW SUBROUTINE
310 REM
320 LET D = INT(10*RND)
330 IF D > 6 THEN GOTO 320
340 IF D < 1 THEN GOTO 320
350 RETURN
360 REM
370 REM
380 REM
390 REM  SELECT SUBROUTINE
400 REM
410 IF D = 1 THEN GOSUB 530
420 IF D = 2 THEN GOSUB 600
430 IF D = 3 THEN GOSUB 670
440 IF D = 4 THEN GOSUB 740
450 IF D = 5 THEN GOSUB 810
460 IF D = 6 THEN GOSUB 880
470 RETURN
480 REM
490 REM
500 REM
510 'GRAPHICS' SUBROUTINES
520 REM
530 PRINT "         "
540 PRINT "|_______|"
550 PRINT "|       |"
560 PRINT "|   *   |"
570 PRINT "|       |"
580 PRINT "|_______|"
590 RETURN
600 PRINT "         "
610 PRINT "|_______|"
620 PRINT "|     * |"
630 PRINT "|       |"
640 PRINT "| *     |"
650 PRINT "|_______|"
660 RETURN
670 PRINT "         "
680 PRINT "|_______|"
690 PRINT "|     * |"
700 PRINT "|   *   |"
710 PRINT "| *     |"
720 PRINT "|_______|"
730 RETURN
740 PRINT "         "
750 PRINT "|_______|"
760 PRINT "| *   * |"
770 PRINT "|       |"
780 PRINT "| *   * |"
790 PRINT "|_______|"
800 RETURN
810 PRINT "         "
820 PRINT "|_______|"
830 PRINT "| *   * |"
840 PRINT "|   *   |"
850 PRINT "| *   * |"
860 PRINT "|_______|"
870 RETURN
880 PRINT "         "
890 PRINT "|_______|"
900 PRINT "| *   * |"
910 PRINT "| *   * |"
920 PRINT "| *   * |"
930 PRINT "|_______|"
940 RETURN
```

```
 5 LET R = 0
10 PRINT "HIT THE SPACE-BAR"
20 FOR X = 0 TO 1
30 LET R = R + 1
40 IF R > 9 THEN LET R = 1
50 IF INKEY$ = "  " THEN GOTO 80
60 LET X = 0
70 NEXT X
80 PRINT R
```

In this program, R can never be less than 1 or greater than 9. By the time the space bar is pressed (and recognised by the INKEY$ function in line 50), R will have a value somewhere between 1 and 9 inclusive.

This program was tested 1,000 times and found an average value for R of 5.014. Since a perfect average would be 5 and the error is only 0.28% high, this suggests that the program does indeed generate a random number very close to the theoretical average. The point is, of course, that even when a program appears reasonable on paper, there may be unforeseen flaws in it. Actual testing is well worth while.

Some readers will have noticed that these random number programs could be shortened by using various GOTO statements in place of the FOR . . . NEXT loop. Our reason for avoiding GOTO statements will become clearer in future parts of the Basic Programming course.

Basic Flavours

On the BBC Micro and the Oric-1, delete line 20, and replace line 320 by:

```
320 LET D = INT(10*RND(1))
```

On the Dragon-32, delete line 20, and replace line 320 by:

```
320 LET D = RND(6)
```

and delete lines 330 and 340.

On the Lynx, replace line 20 by:

```
20 RANDOM
```

On the Vic-20 and the Commodore 64, replace line 20 by:

```
20 LET X = RND(-TI)
```

and replace line 320 by:

```
320 LET D = INT(10*RND(1))
```

On the Oric-1 and the Lynx, replace INKEY$ by KEY$

On the Vic-20 and the Commodore 64, replace line 40 on page 173 by:

```
40 GET A$
```

and replace line 50 on page 175 by:

```
50 GET A$: IF A$ = "  " THEN GOTO 80
```

On the ZX81, modify lines 10 and 50 in the programs on pages 173 and 175 by replacing the space (" ") with "X"

On the BBC Micro, replace INKEY$ by INKEY$(10). The number in brackets is the time in hundredths of a second during which the system will wait for a keypress

Exercises

■ **RND Function** Modify the last program in the text to give a random number in the range 1 to 6 (inclusive).

■ **Loop And Average** Add lines to the last program in the text to make it repeat 100 times and produce an average of the 100 results.

■ **Replace With Subroutine** Replace lines 50 and 130 in the main program (the random dice throw subroutine) with a GOSUB calling your 'random number generator' in the first exercise.

■ **INKEY$** Using the INKEY$ function, how would you write a program to read any key typed at the keyboard and print: THE KEY YOU HIT WAS: * as a result (* represents the key you pressed).

■ **Timing Loop** Write a timing loop (a 'counting' loop) and use the INKEY$ function to find how big the value of a variable becomes after 10 seconds (you'll need to use a watch). Write the program so that the final printout reads: THE VALUE OF R AFTER 10 SECONDS IS: * (* represents the value of R).

■ **IF-THEN Tests** Write a simple game program in which the computer generates a random number between 1 and 100 (inclusive) and the player has to guess what the number is. The player has five tries. Each time, the program responds with the messages YOUR GUESS IS TOO LARGE, YOUR GUESS IS TOO SMALL, or YOU ARE RIGHT, CONGRATULATIONS, or NO MORE GOES. YOU LOSE!

Answers in the next issue.

Answers To 'Exercises' On pages 148-149

Loops 1
THE VALUE OF A IS 450
Loops 2
START
STOP
Loops 3
THE VALUE OF A IS NOW 160
Loops 4
I LIKE BASIC
I LIKE BASIC
I LIKE BASIC
:
:
Until you RESET or BREAK the program
Loops 5
I'M FEELING LOOPY
(15 times)
Read-Data 1
WE'RE TESTING THE READ STATEMENT
170
Read-Data 2
X=1
X=2
:
:
X=23

On Two Wheels

Robot technology is developing fast. Floor robots such as the BBC Buggy can even be programmed to detect obstacles

Floor robots and turtles are more than just educational, they are fun! The principles involved in controlling a device like the BBC Buggy are the same as those needed for full-sized industrial robots. Although they won't actually do the housework yet, they could become the next generation of domestic aids.

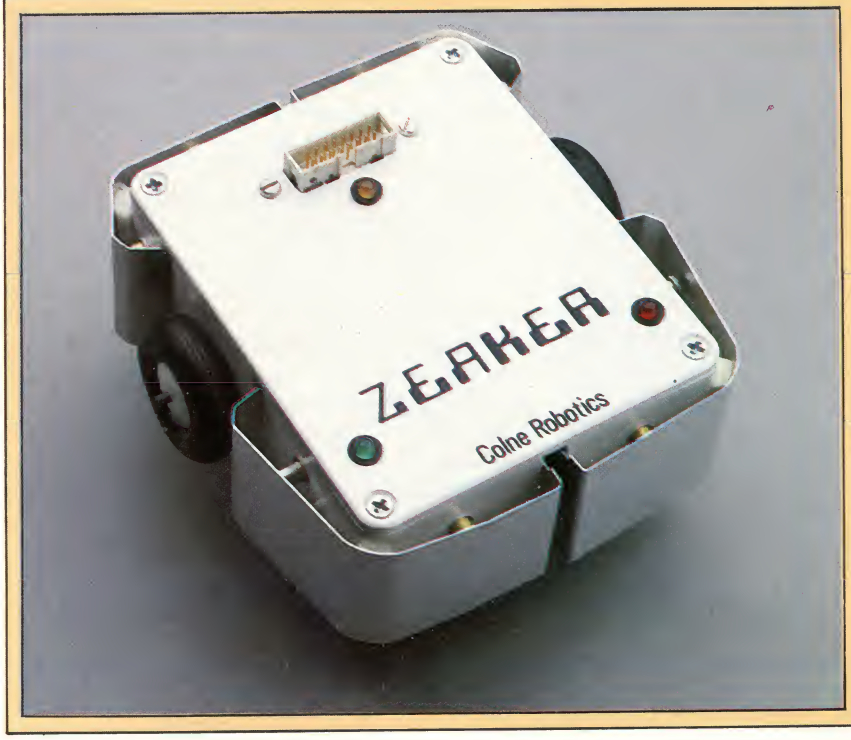
Robots need to be capable of being precisely positioned in relation to their surroundings and for this reason are usually driven by stepper motors. Unlike conventional motors, stepper motors do not rotate when power is applied. The spindle turns by a predetermined fraction of a complete rotation for each pulse of power that is applied. The number of pulses needed to make a complete revolution depends on the particular motor. The direction of rotation can also be controlled. It is possible to make a floor robot or turtle move in very precise distances, in any direction, by letting the computer drive the motors separately. A floor robot can turn on the spot simply by driving the two wheels in opposite directions.

However, it is equally important for the robot to be able to report back to the computer when it encounters something. Collisions are usually detected by mounting bumpers around the body of the vehicle, which are connected to microswitches. These, in turn, are connected back to the computer's input port and the closing or opening of each switch will cause one of the bits to change between '0' and '1'.

Other sorts of input from the robot are often required. An ability to follow white lines on a black floor may be useful. This is achieved by arranging a light source on the robot to shine down, with a detector next to it to measure the amount of light reflected back. This amount will vary according to the surface that the robot is passing over at the time; it is an analogue quantity rather than a digital one. The BBC Micro is fitted with an analogue input which enables a detector of this sort to be used directly. Most other systems will have to convert the signal to a digital one before sending it back to the computer.

Another use for this sort of detector is as a bar code reader. Codes relating to the nature of items stored in a warehouse could be scanned as the robot searches for the correct item. The BBC Buggy is supplied with some demonstration software that uses its bar code reader to play music; the principles are the same.

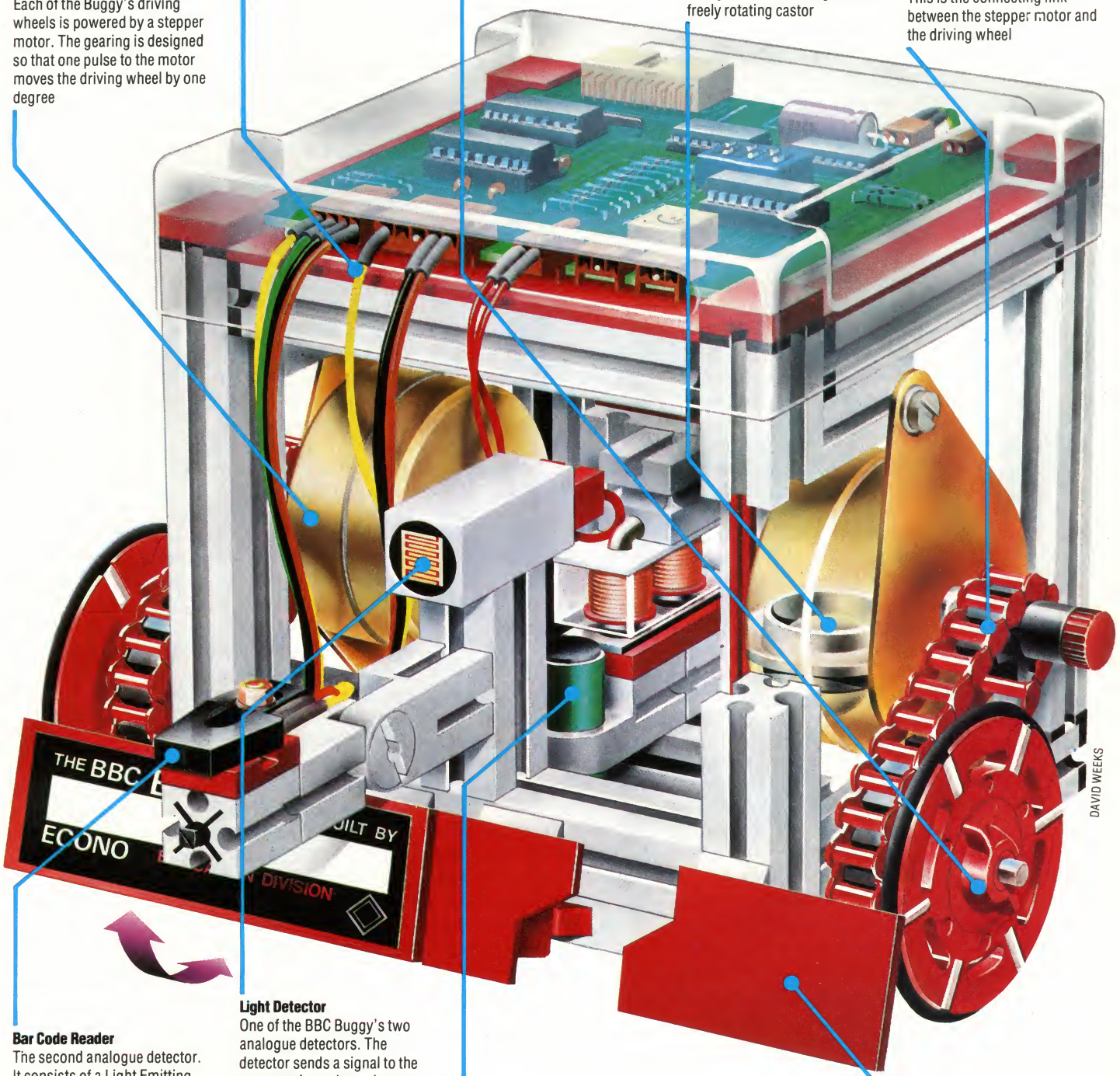
Other kinds of analogue signal that the floor robot might like to follow are light, sound and



CHRIS STEVENS

On Course

This 'micro-turtle' is really halfway between a floor robot and a purpose-designed turtle in that it possesses collision detectors and so can be given a degree of 'intelligence'. The Zeaker is capable of moving forwards, backwards, left and right and moving a pen up and down. Control is made very simple by the provision of a version of the LOGO language called 'SNAIL LOGO', so commands such as FORWARD and BACKWARD can be used directly. Computers other than the BBC Micro will require an interface unit. This is an extra expense, but the interface unit does have its own power unit and therefore doesn't place any strain on the computer's power supply



Driving Wheel
The rubber-tyred wheel on each side of the Buggy provides the movement, and stops the device from slipping, so that accurate movements can be made

Control Board
All the signals to and from the computer are processed by this interface card before being relayed

Trailing Castor
The rear of the Buggy's triangular undercarriage is this freely rotating castor

Driving Chain
This is the connecting link between the stepper motor and the driving wheel

Stepper Motor
Each of the Buggy's driving wheels is powered by a stepper motor. The gearing is designed so that one pulse to the motor moves the driving wheel by one degree

Bar Code Reader
The second analogue detector. It consists of a Light Emitting Diode (LED) and a Light Detecting Diode (LDD). Light from the LED is reflected from the surface under the Buggy and detected by the LDD. The amount of light reflected depends on the material and its colour, so as well as reading bar codes, the detector can also be used for following lines on the floor

Light Detector
One of the BBC Buggy's two analogue detectors. The detector sends a signal to the computer's analogue input port that is proportional to the amount of light detected — the more light shining on the Buggy, the bigger the signal

Pen Holder
The BBC Buggy can be fitted with an optional pen holder between the wheels to enable it to draw lines on the surface it crosses. The holder is moved up and down under the control of the computer

Front Collision Detectors
Each side of the split front bumper is fitted with a microswitch that gives an on (collision) or off (no collision) signal to the computer

magnetic fields. These fields are often used in industrial applications where a robot must follow a fixed path through a warehouse or factory floor. Special cables are buried in the floor which leave a magnetic 'path' for the robot to follow.

Control of the floor robot is generally handled through a set of specially-written program routines. These handle the sending and receiving of information over the cable link from the computer's user port to the device. In the case of the BBC Buggy four bits are used to control the motors. Data is sent from the Buggy along the same cable. The analogue outputs from both the light sensor and the bar code reader go to the analogue port, and the two collision detectors are connected to two other input lines on the user port.

An I/O port such as the user port can be examined by looking at a certain location in the computer's memory map. The BASIC command PEEK is generally used to read the current contents of the location. To alter the contents, in order to change the direction of one of the motors, and turn the robot, for example, the programmer must alter the value of the appropriate bit in the location. The command POKE will achieve this in a BASIC program.

The analogue information can be examined in much the same way, provided the computer being used has an analogue-to-digital converter built in. If no such device is available then an interface unit must be added to the robot which converts the analogue signals to digital information before sending them back to the computer.

A turtle is really a form of floor robot intended for use in conjunction with the LOGO language, though the distinction between a robot and a turtle is becoming somewhat blurred (see page 164). Many of the latest turtles are equipped with collision detectors, just as many of the floor robots are being fitted with pen holders to enable them to act as turtles. The idea of being able to create large-scale drawings by moving a wheeled pen over the surface originates from the teaching of the relationship between distance, angle and shape. If a person moves 10 units forward, turns left, moves 10 units, turns left, moves 10 units, turns left and moves another 10 units, he will have walked around a square. To illustrate these shapes and the relationship they have with movement we can attach a pen to the robot or turtle and create the shape on paper.

The BBC Buggy comes in the form of a construction kit, so you have to assemble it before you can explore the world of robotics. The Buggy is based on the commercially available Fischer Technik system and so can be expanded and enhanced very simply.

Putting together a device like the Buggy is an education in itself. A lot can be learnt from examining the way the various pieces go together. However, the real learning begins when the user tries to take control of his new 'toy'. Although many of the commercial floor robots come with controlling software it is much more fun to write your own. A completely new approach to programming is required — that of control.

DAVID WEEKS

Dedicated Robot
A turtle is a dedicated form of robot which draws on the floor with a retractable felt-tip pen under the instruction of the computer. Turtles are usually associated with the LOGO educational language (see page 164), though they can be driven by BASIC

While the robot is active the program must constantly monitor its sensors to see whether it has found a line on the floor, detected a strong light source, bumped into a chair and so on. The instant the detector signals something the computer must react to protect the robot from potential damage. Programs of this type are typically called 'real-time' because their responses must be immediate.

In theory there is little difference between a program that can allow a floor robot to roam around a room without hitting objects and one that can control a power plant. The techniques learnt by playing with devices such as the BBC Buggy can also develop an understanding of artificial intelligence. Programs can be written that allow a floor robot to carry out a pre-determined task until a detector senses that its batteries are running low. The robot then searches for a suitable power supply to recharge itself, so that it can carry on functioning.

The next generation of floor robots will offer even more remarkable facilities. They will probably be equipped with grab arms to allow them to fetch and carry small loads. Light sensors may also be replaced by miniature solid-state cameras that will allow the robot to 'see' where it is going. Speech synthesis units will allow the robot to communicate with its operator and speech recognition will open a new channel of control over the robot's actions.

Sweet Sixteen

Computers work in binary notation, since they can cope with only two states. So why do programmers sometimes use hexadecimal, which is base 16?

It is easy to see why the binary system is used internally in computers — the representation of numbers using just ones and zeros lends itself well to the on/off electrical signals of the computer. It is also easy to see why the decimal system is almost universally used in human society — we happen to have ten fingers and so it has become our natural number base.

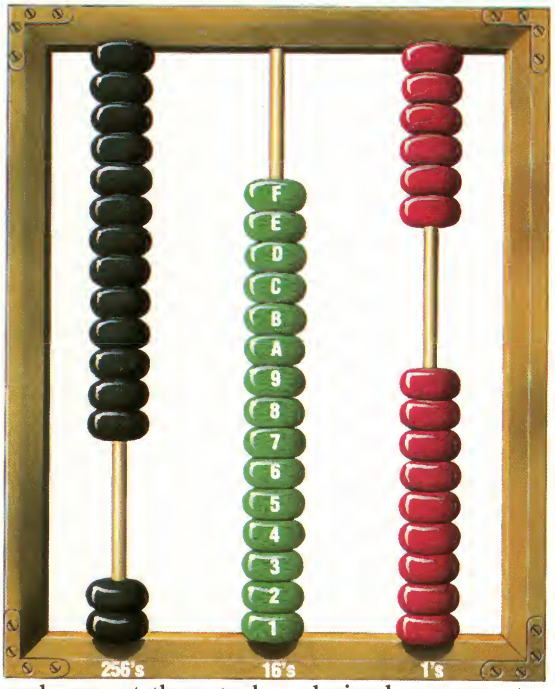
But why should hexadecimal (base 16) have any use? The reason is that hexadecimal numbers convert to and from binary numbers much more easily than decimal numbers do. Since computers 'understand' binary numbers, why don't programmers program in binary digits? There are two reasons. Firstly, binary numbers are much longer than their base 10 or base 16 equivalents. For example, the number 356 in decimal is 0000000101100100 in binary. Secondly, it is much easier to make mistakes using numbers comprising only ones and zeros.

The hexadecimal system uses 16 different digits, including zero, so that the largest number that can be represented in the 'ones' column is equivalent to decimal 15. Mathematicians could have invented six new symbols for the numbers between ten and 15, but the convention is to use the first six letters of the alphabet for the extra six digits. When we count by adding ones, we use the standard numeric digits together with the alphabetic digits until we reach F — the hexadecimal equivalent of 15. After that we run out of symbols and have to start a new column, the 'sixteens' column. Here, too, we can use the available symbols up to F. When both the 'sixteens' column and the 'ones' column are full, adding one means we have to start a third column, the 'two-hundred-and-fifty-sixes' column. The panel shows some decimal numbers with their binary and hexadecimal equivalents.

The number 65,535 represented in hexadecimal is FFFF, so one of the advantages of the hexadecimal system is immediately apparent. A number that takes five digits to represent in decimal and 16 to represent in binary needs only four digits in hexadecimal. Apart from this compactness, there is a more important advantage. Four binary digits can be represented by exactly one hexadecimal digit. This makes converting from binary to hexadecimal and vice versa relatively straightforward.

To convert a binary number into its hexadecimal equivalent, divide the binary number into groups of four digits, starting from the right,

MARK WATKINSON



The Hexadecimal Abacus
An abacus made for the hexadecimal system would have 15 beads on each rod. On the rightmost rod, beads can be flicked down one at a time until the last bead, F, is reached. Hexadecimal is a base-16 number system and letters of the alphabet are used for the numbers higher than 9. As shown on the middle rod, A corresponds to decimal 10, B to 11, C to 12, D to 13, E to 14 and F to 15. After the 'number' F, we run out of digits and have to move over to the next column, the '16's' column .

The '1's' column in the illustration shows a count of 9 beads. The '16's' column shows a count of F and the '256's' column shows a count of 2. This is read as '2F9 hex'. It is equivalent to the decimal number 761 (256 × 2 + 16 × 15 + 1 × 9)

and convert them to hexadecimal a group at a time. Here are some examples:

		1110	1001	binary
		=E	9	hex
		=233		decimal
	1111	1000	1100	binary
	=F	8	C	hex
	=3980			decimal
0111	1110	0010	1101	binary
=7	E	2	D	hex
=32,301				decimal

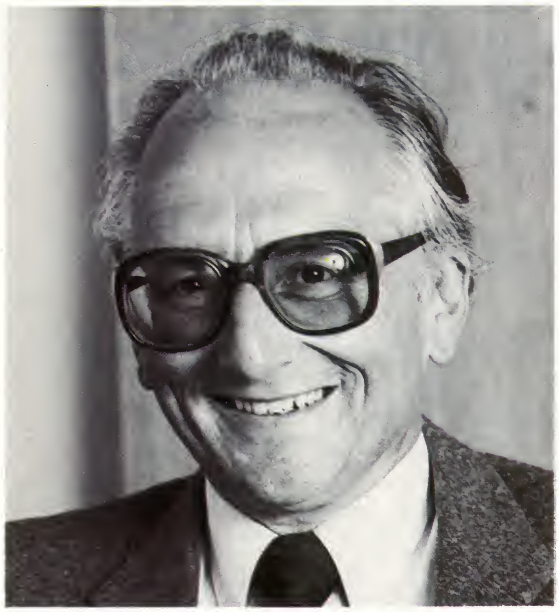
When would you actually want to use hexadecimal notation? Although high-level languages such as BASIC do not require you to use hexadecimal (or binary for that matter), low level languages such as machine code and assembly language do.

Numbers written in hexadecimal are often written with an H after them to distinguish them from decimal numbers. Thus, 256 (decimal) would be written 100H and pronounced 'one zero zero hex'.

We won't be making much practical use of these hexadecimal numbers in THE HOME COMPUTER COURSE. However, you may well come across the term in appendices to your computer's instruction manual. Now, you can at least work out their decimal or binary equivalent.

Binary	Decimal	Hex
00000001	1	1
00000010	2	2
00000011	3	3
00000100	4	4
00000101	5	5
00000110	6	6
00000111	7	7
00001000	8	8
00001001	9	9
00001010	10	A
00001011	11	B
00001100	12	C
00001101	13	D
00001110	14	E
00001111	15	F
00010000	16	10
00010001	17	11
00010010	18	12
00010011	19	13
00010100	20	14
00010101	21	15

Chuck Peddle



The brain behind the 6502 microprocessor, now mass-produced and used in most personal computers

A whole generation older than 'whizz-kid' entrepreneurs like Steve Wozniak and Steve Jobs, Chuck Peddle's first involvement with microprocessors was in 1973, when he joined Motorola to work on the 6800 microprocessor design project.

Since it was one of the first microprocessors on the market, Motorola was able to demand a high price for the 6800 — $200 per unit. Peddle considered the product to be vastly over-priced and left Motorola to join MOS Technology.

At this relatively small company he went to work on another microprocessor design project that was to become the 6502 MPU, arguably the most successful microprocessor in the first decade of microcomputing. At the time, however, no one realised that the product on which they were working was destined to become the mainstay of an entire industry, and contribute in great measure to a social revolution the like of which the world has not seen for 200 years.

One of the few people to grasp the significance of the microprocessor in general, and the potential of the MOS Technology 6502 in particular, was Jack Tramiel, Commodore's President. Until that time, Commodore Business Machines had been involved in a range of office products and pocket calculators, without significant success.

Commodore was MOS Technology's chief customer, regularly purchasing large numbers of dedicated four-function calculator chips. Tramiel, despite his own difficulties in keeping Commodore's head above water, had enough faith in the 6502 to find the money from somewhere to buy out MOS Technology. With the company he acquired the services of Charles Peddle, now Microprocessor Development Engineer.

By this time, Peddle had realised that his brainchild could perhaps power a conceptual breakthrough — the personal computer. The same idea was being pursued independently by Wozniak and Jobs at Apple Computer, Inc. So concerned was Peddle that the new technology should be properly used that he got together with Bill Gates, founder of Microsoft — famous for its BASIC interpreter — in an attempt to buy Apple, which coincidentally came up for sale at the same time as MOS Technology. However, Wozniak and Jobs were asking $150,000 for the company, and Peddle and Gates could come up with only two-thirds of that amount.

Peddle stayed with Commodore and took on the task of originating the Commodore PET (Personal Electronic Transactor). It was launched in 1977, at roughly the same time as the Apple II. The PET was different in that it had a built-in monitor and cassette deck, and the 'feel' of the keyboard was closer to that of a calculator than a typewriter. Within a short time of its debut Commodore had secured orders for a thousand units (at £695 per unit), and the first generation of microcomputers designed specifically for use in the home was born.

It was three years before Peddle was to realise his second major ambition — that of running his own computer company. With Chris Fish, one of the financial brains behind Commodore's sudden rise, he joined Victor United, a subsidiary of the giant Walter Kidde Corporation, and started Sirius Systems Technology.

Development work in the personal computer industry was firmly concentrated on 16-bit chips like Intel's 8088. IBM, it turned out, was also working on a desk-top personal computer based on the same chip, but by chance Sirius was able to present the fruits of its labours just a few weeks earlier. The machine won wide acclaim, and soon established a strong presence in the market, being the first mass-produced, low-cost microcomputer to offer the advantages of the new generation of 16-bit microprocessors.

The Sirius 1 was relatively cheap and easy to use. With its detachable keyboard, high-resolution graphics and anti-glare screen, it set new standards for office microsystems. Users found the vastly enhanced speed and addressing capabilities of the 16-bit microprocessor tremendously beneficial.

All in all, Chuck Peddle had gone a long way towards fulfilling his ambition — to bring computing power within reach of all. And in the process he set a standard for others to follow.

COURTESY OF SIRIUS COMPUTING

The First Personal Computer After developing the 6502 microprocessor, Chuck Peddle set about designing something called a 'personal computer' that was completely self-contained, could be plugged into the mains and used immediately, for whatever purpose the user required. The resulting Commodore PET, which appeared at almost the same time as Steve Wozniak's Apple II, featured a built-in screen, cassette recorder and Microsoft BASIC interpreter. Although it has since gone through several re-designs and face lifts, such as the full-size keyboard pictured, the PET is still popular. One of the early machine's most appealing features was the physical design which subtly suggested a head and shoulders. The 6502 is now the most widely used microprocessor in home computers

Seen In Perspective

Have you ever wondered why certain arcade games such as Asteroids and Battle Zone have such sharp, crisp screen displays compared with those generated by most home computers?

How are those smooth zoom-effects and perspective drawings achieved? Producing them on a home computer is no easy matter, even with plenty of experience. If it were, you'd see more programs that use these techniques. There are very few, and those that exist are not as impressive.

These arcade effects are possible because of the method used to generate the display. This method differs from that used in home computers, not only in the electronics and the programs used, but also in the construction of the display tube itself.

As explained in the article on monitors (see page 132), the picture on a television or monitor tube is produced by moving an electron beam across the phosphor in a series of lines, turning it on and off to produce dots of varying intensities. An image on the screen, therefore, is made up of small dots that lie on a series of lines, like beads on a string. This system is known as 'raster graphics' after the series of lines ('rasters') drawn on the screen by the beam.

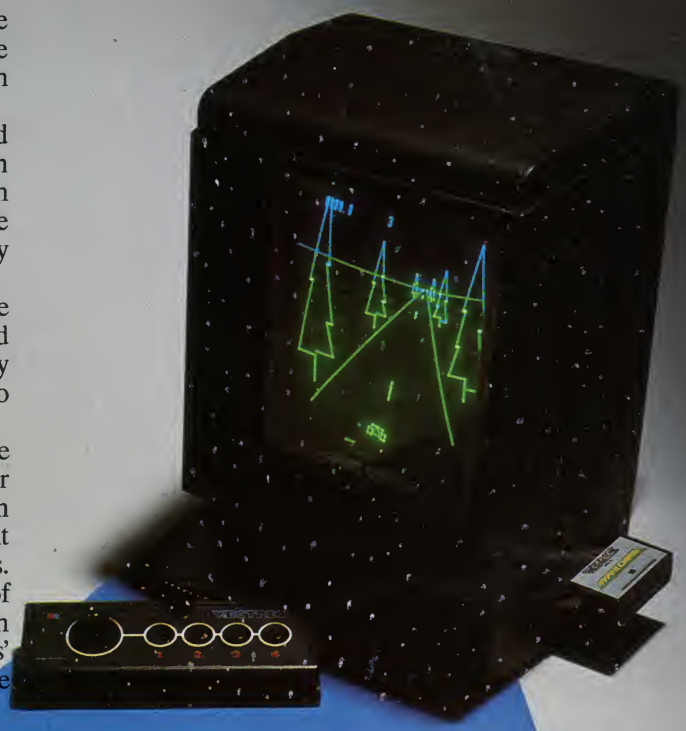
As you will notice next time you go into an arcade, the display in the Asteroids game seems to be made up of 'wires', with no solid areas and no shading. Everything that makes up the image is either black or bright, whatever the colour.

This is the clue to the fact that these arcade machines use a different principle to produce the image. Instead of starting with a dot at the top left-hand corner of the screen and building up the picture in lines, they move the dot only to where it is required. Thus, if a square is needed the beam describes a square shape. Making the square bigger or smaller is simply a case of applying a larger or smaller voltage to the deflection coils around the neck of the tube, which are the magnets that actually direct the beam.

Multiple shapes are almost as simple since, after drawing the first one, the beam is simply turned off, moved to a new position and turned on again, and the moves are repeated. Once all the objects have been drawn, the beam is moved back to the start of the first shape and the process repeated.

Since the screen is flat there are only two dimensions in which the beam can be moved. However, convincing three-dimensional effects can be achieved. These effects rely on the same principle but creating them is a more complex process requiring fast trigonometrical calculations.

By varying the values for the angle of view, the object can be made to tilt, rotate, approach the viewer and retreat, and perform other gyrations with satisfying smoothness. There's a side-effect, though, which is caused by the fact that it is difficult to work out when a line should be hidden. Indeed, it is so difficult that it normally isn't attempted, and all the objects seem to be transparent. They look as though they are made of wires — hence the term 'wire-frame graphics'.



IAN McKINNELL

The object is drawn as a series of 'vectors'. A vector is a line with a specified length and direction, and the system is known as 'vector graphics', distinguishing it from the raster graphics system used in televisions and monitors.

How can you achieve these effects at home without buying your own arcade machine? Not long ago this was impossible but now a domestic video games machine called Vectrex is available, which works on the vector graphics principle.

The Vectrex is a specialised console unit with its own dedicated screen and joystick unit. It runs on cartridges, much like any games console, and most of the popular vector graphic games are available on it, plus others unique to the Vectrex. The manufacturers, Milton Bradley, say they intend to introduce a full keyboard and a BASIC programming cartridge as an add-on soon. These machines have a major disadvantage, however. They cannot produce shades of colour or solid areas, and colour is limited to one intensity of one colour — usually green or white.

Some arcade games' colour is achieved by placing coloured overlays on top of the monochrome screen to tint the object drawn underneath. The Vectrex system, which reproduces different games depending on the cartridge, includes transparent plastic sheets printed with the colours and details necessary for each game. These are fitted in front of the screen.

Despite the remarkable effects that can be produced, the vector graphics system is unlikely to attain great popularity. It is somewhat restricted, and vastly improved raster graphics systems will eventually be able to produce equivalent effects. In the meantime, although a cloud hangs over the Vectrex since US production ceased, it continues to be made in Britain and to enjoy success.

Hyperchase
The Vectrex is a dedicated video games console. It is the only domestic unit to make use of 'vector graphics' in its inbuilt screen, rather than the 'raster scan' method used by televisions and monitors.

The photograph shows the car racer game Hyperchase. Unusually, Vectrex has a built-in game — Minestorms — which is automatically run unless another games cartridge is plugged in. A full keyboard and BASIC programmability are hoped for in the future

COURTESY OF MILTON BRADLEY LTD

.....THE HOME COMPUTER COURSE .....BINDER.

A BASIC NECESSITY.

To help you keep your copies immaculate we have made a very special binder offer. Details of this offer and a special order form are with this issue.

Overseas readers:
This binder offer applies to readers in the UK, Eire and Australia only. Readers in Australia should complete the special loose insert in Issue 1 and see additional binder information on the inside front cover. Readers in New Zealand and South Africa and some other countries can obtain their binders **now.** For details please see inside the front cover.

Binders may be subject to import duty and/or local tax.

NEXT TO YOUR COMPUTER.....YOUR COURSE MANUALS!